Project 3: Analyses of daily COVID-19 cases across nations

Jiayi Shen (js5354), Siquan Wang (sw3442), Jack Yan (xy2395)

5/1/2020

1. Introduction

The pandemic of COVID-19 is the biggest challenge that the world is facing right now. Our lifes are all deeply affected by this public health crisis. By building a model on the growth of COVID-19 cases, we can have a better understanding of the current status and then plan future responses. Thus analyzing existing data and predicting future trajectories has become the most important task faced by public health expertises and policy makers.

The objective of this project includings:
- Develop an optmization algorithm to fit a logisitc curve to each region, using the global COVID-19 data;
- Evaluate how appropiate it is to use such model;
- Apply clustering methods to the logistic grwoth model parameters, and observe the patterns.

2. Data

We used the latest data updated on 04/29/2020, so that we have more data points. The data recorded the following variables:

Id: Record ID

Province/State: The lcoal state/province of the record; 54% records do not have this info;

Country/Region: The country/regionoof the record;

Lat: Lattudiute of the record;

Long: Longitude of the record;

Date: Date of the record; from Jan 21 to March 23;

ConfirmedCases: The number of confirme case on that day;

Fatalities: The number of death on that day;

For the purpose of fitting logistic growth curve, the main variables of interest are `Country/Region` and `ConfirmedCases`. We groupped the dataset by `Country/Region`, and pulled out `days since first case` and `cumulative confirmed cases on each day`.

3. Method

3.1 Logisitic curves

Logisitic curves could be one way to model the trajectory of cumulative cases; It is a parametric function with the form

$$f(t) = \frac{a}{1 + exp\{-b(t-c)\}},$$

where t is the days since the first infection; a is the upper bound, i.e. the maxium number of cases a region can reach, b is growth rate, and c is the mid-point, where the curve changes from convex to concave; Each curve is uniquely defined by (a, b, c). By design a logistic curve increases exponentially at begining and slows down at the end.

Consider the most commonly used loss function: Mean Squared Error (MSE), which takes the form

$$l = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \frac{a}{1+\exp\{-b(T_i - c)\}})^2$$

Then our goal is to minimize the above loss function with respect to (a, b, c). To do this, we can apply Newton-Raphson algorithm. Let $\boldsymbol{\theta} = (a, b, c)$. Newton's method suggests to update $\boldsymbol{\theta}$ iteratively, such that the ith step is given by

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - [\nabla^2 l(\boldsymbol{\theta}_{i-1})]^{-1}\nabla l(\boldsymbol{\theta}_{i-1})$$

where $\nabla l(\boldsymbol{\theta}_{i-1})$ is the gradient, and $[\nabla^2 l(\boldsymbol{\theta}_{i-1})]^{-1}$ is the Hessian matrix.

In this particular case, we replace the Hessian matrix with an identity matrix to simplify the computation. Step-halfing is also incorporated to control the step size to make sure we have a monotone trend.. Then the ith step of our Newton algorithm is given by

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} + \lambda \boldsymbol{H}_{i-1,p\times p}\nabla l(\boldsymbol{\theta}_{i-1})$$

where $\boldsymbol{H}_{i-1,p\times p} = I_{p\times p}$, $\lambda \in (0, 1)$.

The gradient vector $\nabla l(\boldsymbol{\theta}_{i-1})$ of our loss function is:

$$\begin{pmatrix} \partial l/\partial a \\ \partial l/\partial b \\ \partial l/\partial c \end{pmatrix} = \begin{pmatrix} \frac{2}{n}\sum_{i=1}^{n}(Y_i - \frac{a}{1+\exp\{-b(T_i-c)\}}) \cdot \frac{-1}{1+\exp\{-b(T_i-c)\}} \\ \frac{2}{n}\sum_{i=1}^{n}(Y_i - \frac{a}{1+\exp\{-b(T_i-c)\}}) \cdot \frac{-a(T_i-c)\exp\{-b(T_i-c)\})}{(1+\exp\{-b(T_i-c)\})^2} \\ \frac{2}{n}\sum_{i=1}^{n}(Y_i - \frac{a}{1+\exp\{-b(T_i-c)\}}) \cdot \frac{ab\exp\{-b(T_i-c)\})}{(1+\exp\{-b(T_i-c)\})^2} \end{pmatrix} \tag{1}$$

We set the convergence criteria to be that the difference in MSE of two consecutive iterations is smaller than 10^{-5}, and the maximum iteration number to be 1000.

3.2 Clustering

To understand which countries/regions are similar in terms of the trajectory of COVID-19 cases, we applied two clustering methods, K-means and Guassian mixture model, to group the fitted parameters $(\hat{a}, \hat{b}, \hat{c})$.

The guassian mixture model assumes that $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^p$ are i.i.d. random vectors following a mixture mulitvariate normal distributions with k hidden groups. In this case, $\mathbf{x}_i = (Y_i, T_i)'$ and $(a, b, c) \in \mathbb{R}^3$. And

$$\mathbf{x}_i \sim \begin{cases} N(\boldsymbol{\mu}_1, \Sigma_1), \text{with probability } p_1 \\ N(\boldsymbol{\mu}_2, \Sigma_2), \text{with probability } p_2 \\ \quad\vdots \quad , \quad \vdots \\ N(\boldsymbol{\mu}_k, \Sigma_k), \text{with probability } p_k \end{cases}$$

$$\mathbf{x}_i \sim \begin{cases} N(\boldsymbol{\mu}_1, \Sigma_1), \text{with probability } p_1 \\ N(\boldsymbol{\mu}_2, \Sigma_2), \text{with probability } p_2 \\ \quad\vdots \quad , \quad \vdots \\ N(\boldsymbol{\mu}_k, \Sigma_k), \text{with probability } p_k \end{cases}$$

The density of a multivariate normal $\mathbf{x}_i$ is

$$f(\mathbf{x};\boldsymbol{\mu},\Sigma) = \frac{\exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}))}{\sqrt{(2\pi)^p|\Sigma|}}$$

The observed likelihood of $\{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_n\}$ is

$$L(\theta;\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_n) = \prod_{i=1}^{n}\sum_{j=1}^{k} p_j f(\mathbf{x}_i;\boldsymbol{\mu}_j,\Sigma_j)$$

Let $\mathbf{r}_i = (r_{i,1},...,r_{i,k}) \in \mathbb{R}^k$ as the cluster indicator of $\mathbf{x}_i$, which takes form $(0,0,...,0,1,0,0)$ with $r_{i,j} = I\{\mathbf{x}_i \text{ belongs to cluster } j\}$. The cluster indicator $\mathbf{r}_i$ is a latent variable that cannot be observed. Therefore, we use EM algorithm to iteratively estimate model parameters.

E-step: Evaluate the responsibilities using the current parameter values

$$\gamma_{i,k}^{(t)} = P(r_{i,k}=1|\mathbf{x}_i,\theta^{(t)}) = \frac{p_k^{(t)} f(\mathbf{x}_i|\boldsymbol{\mu}_k^{(t)},\Sigma_k^{(t)})}{\sum_{j=1}^{K} p_k^{(t)} f(\mathbf{x}_i|\boldsymbol{\mu}_j^{(t)},\Sigma_j^{(t)})}$$

M-step:
$\theta^{(t+1)} = \arg\max \ell(\mathbf{x},\gamma^{(t)},\theta)$.
Let $n_k = \sum_{i=1}^{n}\gamma_{i,k}$, we have

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{1}{n_k}\sum_{i=1}^{n}\gamma_{i,k}\mathbf{x}_i$$

$$\Sigma_k^{(t+1)} = \frac{1}{n_k}\sum_{i=1}^{n}\gamma_{i,k}(\mathbf{x}_i-\boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}_i-\boldsymbol{\mu}_k^{(t+1)})^T$$

$$p_k^{(t+1)} = \frac{n_k}{n}$$

By iteratively updating the E-step and the M-step, we can reach a solution upon convergence.

On the other hand, the K-means algorithm essentially finds cluster centers and cluster assignments that minimize the objective function

$$J(\mathbf{r},\boldsymbol{\mu}) = \sum_{i=1}^{n}\sum_{j=1}^{k} r_{i,j}\|\mathbf{x}_i-\mu_k\|^2$$

where $\{\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,...,\boldsymbol{\mu}_k\}$ are the centers of the k (unknown) clusters, and $\mathbf{r}_i = (r_{i,1},...,r_{i,k}) \in \mathbb{R}^k$ as the *hard* cluster assignment of $\mathbf{x}_i$. Both Gaussian mixture model and k-means are popular clutsering methods and we would like to compare thier performance in our dataset.

4. Results

4.1 Logistic curve

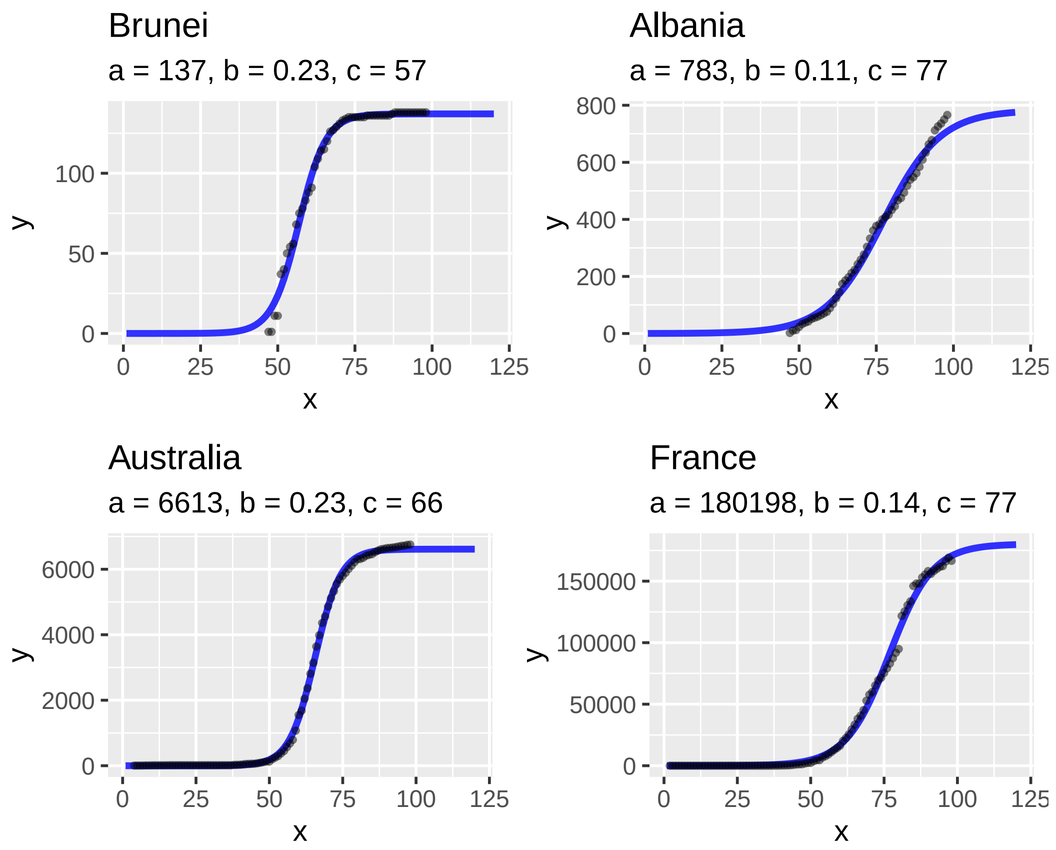


Figure 1 Logistic curves fitted on cumulative confirmed cases in selected countries

Logistic curves were used to fit cumulative confirmed cases in each country/region. The fitted results for some selected countries are shown in **Figure 1**. According to the model, as of 04/29/2020, 37 regions have passed the midpoint, among which 100 regions are close to the end of virus spreading. If the confirmed cases in a region is greater than 95% of the estimated `a` (upper bound) in the logistic curve, we define the region as close to the end of virus spreading.

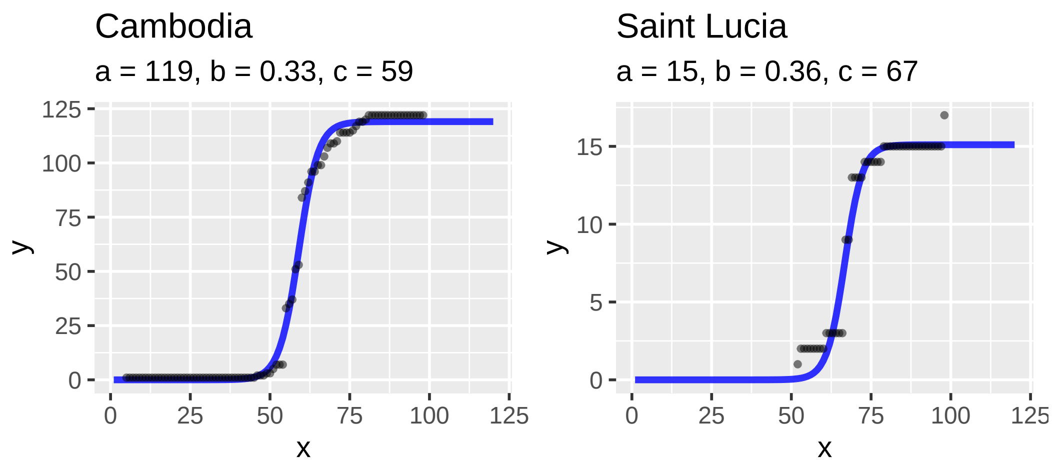


Figure 2 Logistic curves of selected countries with steep rate of growth

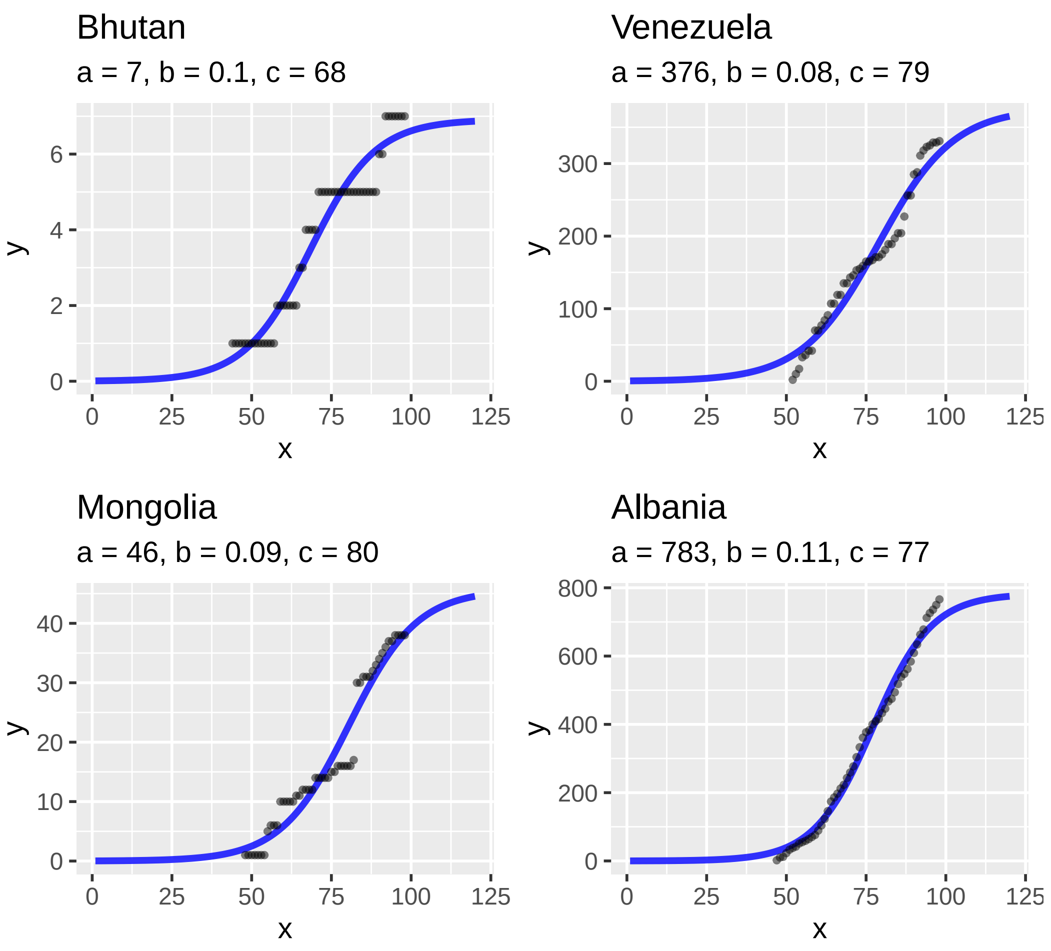


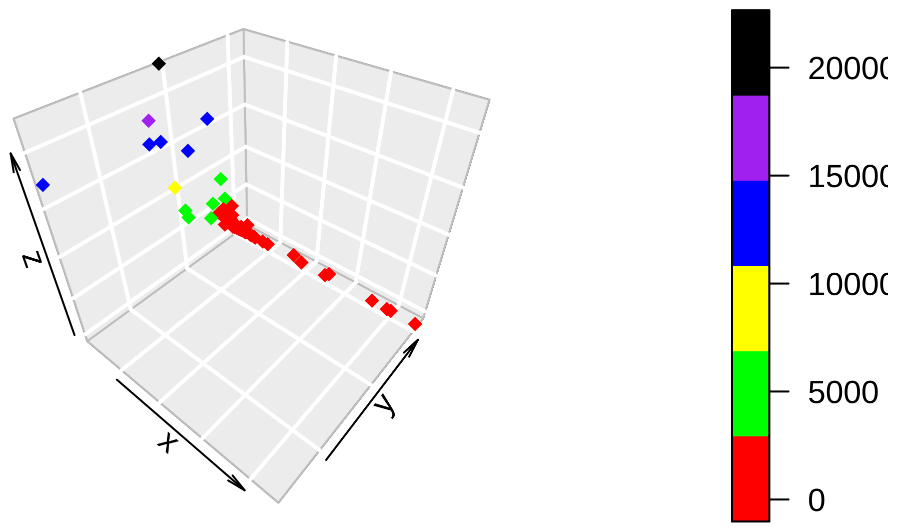
Figure 3 Logistic curves of selected countries with flat rate of growth

In general, some regions that are close to or have reached the end of virus spreading (at least according to the model) tend to have a more steep rate of growth (**Figure 2**). On the other hand, regions with small population size or at an early stage of spreading tend to have a more flat rate of growth, such as Bhutan and Venezuela (**Figure 3**).

Logistic curve might not be an appropriate model for fitting the curmulative cases and predicting future new cases. First our Newton's algorithm suggests that although we spent much efforts in optimizing this optimization algorithm (including standardization, replacing Hessian by Identity mattrix and step-halfing), the convergence speed might still have some unknown internal issue and the results highly depending on the initial starting value if we do not do standardization, which means that our model is not that robust and the estimated curve shape parameters might be questionable in some unusual case. Second, some African countries' results are little bit strange base on our model, which might be due to too liitle data available. Finally, I may suggest using some other exponential-distribution based curves to fit the data and incorporate more shape paramters in the model fitting process. Or we could some hybrid approaches to fit different types of curves based on preclassification of countries.

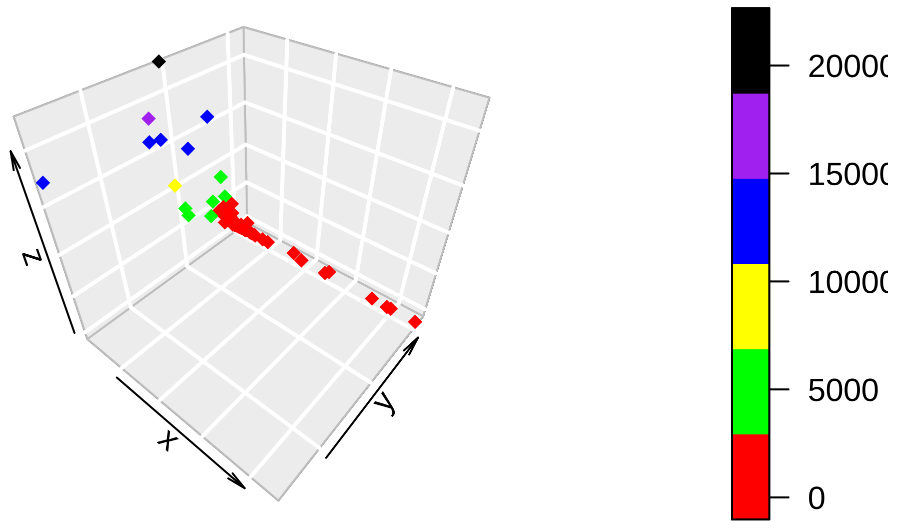
4.2 Clustering fitted curves

We first wrote the function to apply EM algorithm for multivariate Gaussian mixture model. Then we tried different number of mixture components. Code is in the Appendix.



Since we are doing clustering based on three paramters simutaneouly: a, b and c, it will be a little bit messy if we visualize the result in a 3-D plot. So we just simply provided the above result and dertermined which number of mixture components is better. From the above plots we could see that when the number of mixture components is 4, the clustering algorithm gives us a stable result. Too small or too big number of mixtures will either cannot capture the underlying pattern or make the plot more messy.

Similarly we applied the k-means clustering method into our data:



Following similar logic in Gaussian mixture model, we just simply provided the above plots and dertermined which value of k is better. Based on the graph, we conclude that K = 4 gives us the best result, which corresponds to the result of Gaussian mixture model. So both clustering algorithms suggests that we could choose 4 clustering centers in our dataset. As an example of illustration, we would use the result of k-means clustering to answer the questions listed.

In this porject both K-mean and Guassian mixture model (with EM algorithm) give us similar clustering results with 4 clusters as optimal. We could clearly see patterns in the clustering result, for example, some of the most disease-severe countries are clustered together such as : Italy, US, France and Spain. Also some of the South-American geograpahic region countries are clustered togther such as Chile, Brazil. Finally, most of the countries in the world are clustered together as either their disease situation is controlled and thus not that severe, or still under development but the total case is still relatively low.

Appendix

```r
library(tidyverse)
library("plot3D")
library(patchwork)

series_df =
  read_csv("time_series_covid19_confirmed_global_0429.csv") %>%
  select(-Lat, -Long)
date_varname = series_df %>% select(-"Province/State", -"Country/Region") %>% names()

covid =
  series_df %>%
  rename(province_state = "Province/State",
         country_region = "Country/Region") %>%
  group_by(country_region) %>%
  summarize_at(date_varname, sum) %>%
  gather(key = "date", value = "confirmed_cases", "1/22/20":"4/29/20") %>%
  filter(confirmed_cases != 0) %>%
  mutate(date = paste0(date, "20"),
         date = as.Date(date, "%m/%d/%Y")) %>%
  arrange(country_region, date) %>%
  mutate(earliest = min(date),
         days = as.integer(date - earliest))

# saveRDS(covid, "covid.RDS")

place_list = unique(covid$country_region)

# Newton-Raphson
# difference between beta1 and beta2
beta_norm = function(beta1, beta2) {
  # beta1 and beta2 are p*1 vectors
  norm(as.matrix(beta1 - beta2), "F")
}

# loglik
logisticstuff <- function(X, Y, a, b, c) {
  estuff = exp(-b*(X-c))

  # sum of squares at current value
  r <- Y - a/(1+estuff)
  sumsq <- sum(r^2)

  da = sum(r*(-1/(1+estuff)))
  db = sum(r*(-a*(X-c)*estuff/(1+estuff)^2))
  dc = sum(r*(a*b*estuff/(1+estuff)^2))

  # gradient at betavec
  grad = c(da, db, dc)

  Hess = diag(3)/10
  return(list(sumsq = sumsq, grad = grad, Hess = Hess))
}

# newton raphson
newton = function(X, Y, func = logisticstuff, start, tol = 1000, maxiter = 200) {
  i <- 0
  cur <- start
  stuff <- func(X, Y, cur[1], cur[2], cur[3])
  res <- c(0, stuff$sumsq, cur)
  prevsumsq <- 0
  prev = rep(0,3)
  while(i < maxiter && abs(stuff$sumsq - prevsumsq) > tol) {
    i <- i + 1
    prevsumsq <- stuff$sumsq
    prev <- cur
    Hess = stuff$Hess
    step = 1
    cur <- prev - solve(Hess) %*% stuff$grad * step
    stuff_temp <- func(X, Y, a = cur[1], b = cur[2], c = cur[3]) # log-lik, gradient, Hessian
    k = 1
    while (stuff_temp$sumsq > prevsumsq & k<100) {
      k = k + 1
      step = step/2
      cur <- prev - solve(Hess) %*% stuff$grad * step
      stuff_temp <- func(X, Y, a = cur[1], b = cur[2], c = cur[3]) # log-lik, gradient, Hessian
    }
    stuff = func(X, Y, cur[1], cur[2], cur[3])
    res <- rbind(res, c(i, stuff$sumsq, cur[1], cur[2], cur[3]))
  }

  return(res)
}

# Apply Newton to all countries
logisticf = function(X, a, b, c) {
  estuff = exp(-b*(X-c))
  return(a/(1+estuff))
}

logit_growth = function(place){

  Y = filter(covid, country_region == place) %>% pull(confirmed_cases)
  X = filter(covid, country_region == place) %>% pull(days)

  digits_Y  = max(nchar(trunc(Y)))-2
  a_start = max(Y/(10**(digits_Y)))
  b_start = 1
  c_start = max(median(X/10), 5)
  re = newton(X = X/10, Y = Y/(10**digits_Y), func = logisticstuff,
              start = c(a_start,b_start,c_start),
              tol = 1e-10, maxiter = 1000)
  return(list(a = re[nrow(re),3]*(10**(digits_Y)),
              b = re[nrow(re),4]/10,
               c = re[nrow(re),5]*10))

}


res_df = data.frame(country= NULL, a = NULL, b = NULL, c = NULL)
for (place in place_list){
  res = logit_growth(place)
  res_df = bind_rows(res_df, bind_cols(country = place,
                               a = res$a,
                               b = res$b,
                               c = res$c))
}
typeof(res_df)
saveRDS(res_df, "logistic_res.RDS")

# function to apply EM algorithm for multivariate Gaussian mixture model
EM_MG_algrm <- function(data, ncluster){

  #setting
  data <- as.matrix(data) %>% scale()
  N <- nrow(data)
  q <- ncol(data)
  p_j <- rep(1/ncluster, ncluster)
  mu <-  data[sample(N, ncluster),  ] %>% as.matrix()
  covmat <- diag(ncol(data))

  covList <- list()
  for(i in 1:ncluster){
    covList[[i]] <- covmat
  }

  count=1
  while(count <100){
    mu0 <- mu

    # E-step: Evaluate posterior probability, gamma
    gamma <- c()
    for(j in 1:ncluster){
      gamma2 <- apply(data,1, mvtnorm::dmvnorm, mean = mu[j,], sigma = covList[[j]])
      gamma <- cbind(gamma, gamma2)
    }

    # M- step: Calculate mu
    tempmat <- matrix(rep(p_j,N),nrow=N,byrow = T)
    r <- (gamma * tempmat) / rowSums(gamma * tempmat)
    mu <- t(r) %*% data / colSums(r)

    # M- step: Calculate Sigma and p
    for(j in 1:ncluster){
      sigma <- matrix(rep(0,q^2),ncol=q)
      for(i in 1:N){
        sigma = sigma + r[i,j] * (data[i,]-mu0[j,]) %*% t(data[i,]-mu0[j,])
      }
      covList[[j]] <- sigma/sum(r[,j])
    }
    p_j <- colSums(r)/N
    count = count + 1
  }

  cluster <- which(r == apply(r, 1, max), arr.ind = T)
  cluster <- cluster[order(cluster[,1]),]
  return(list(mu = mu,covList = covList, p_j = p_j,cluster = cluster))
}

set.seed(8160)
countryname <- res_df$country

model_1 <- EM_MG_algrm(res_df[,c("a", "b", "c")], 3)
model_2 <- EM_MG_algrm(res_df[,c("a", "b", "c")], 4)
model_3 <- EM_MG_algrm(res_df[,c("a", "b", "c")], 5)
model_4 <- EM_MG_algrm(res_df[,c("a", "b", "c")], 6)
cluster_res = list(model_1 = model_1, model_2 = model_2, model_3 = model_3, model_4 = model_4)
saveRDS(cluster_res, "cluster_res.RDS")

cluster_res = readRDS("cluster_res.RDS")
model_1 = cluster_res$model_1
model_2 = cluster_res$model_2
model_3 = cluster_res$model_3
model_4 = cluster_res$model_4
em_result_1 <- cbind(res_df, model_1$cluster[,2])
em_result_2 <- cbind(res_df, model_2$cluster[,2])
em_result_3 <- cbind(res_df, model_3$cluster[,2])
em_result_4 <- cbind(res_df, model_4$cluster[,2])

scatter3D(em_result_1$a, em_result_1$b, em_result_1$c, bty = "g", pch = 18,
          col.var = as.integer(model_1$cluster[,2]),
          col = c("red", "green", "yellow"))
scatter3D(em_result_2$a, em_result_2$b, em_result_2$c, bty = "g", pch = 18,
          col.var = as.integer(model_2$cluster[,2]),
          col = c("red", "green", "yellow", "blue"))
scatter3D(em_result_3$a, em_result_3$b, em_result_3$c, bty = "g", pch = 18,
          col.var = as.integer(model_3$cluster[,2]),
          col = c("red", "green", "yellow","blue", "purple"))
scatter3D(em_result_4$a, em_result_4$b, em_result_4$c, bty = "g", pch = 18,
          col.var = as.integer(model_4$cluster[,2]),
          col = c("red", "green", "yellow","blue", "purple", "black"))


set.seed(8160)
kmeans_model_1 <- kmeans(res_df[,c("a", "b", "c")], 3)
kmeans_model_2 <- kmeans(res_df[,c("a", "b", "c")], 4)
kmeans_model_3 <- kmeans(res_df[,c("a", "b", "c")], 5)
kmeans_model_4 <- kmeans(res_df[,c("a", "b", "c")], 6)
kmeans_result_1 <- cbind(res_df, kmeans_model_1$cluster)
kmeans_result_2 <- cbind(res_df, kmeans_model_2$cluster)
```

```r
kmeans_result_3 <- cbind(res_df, kmeans_model_3$cluster)
kmeans_result_4 <- cbind(res_df, kmeans_model_4$cluster)

scatter3D(kmeans_result_1$a, kmeans_result_1$b, kmeans_result_3$c, bty = "g", pch = 18,
          col.var = as.integer(kmeans_model_1$cluster),
          col = c("red", "green", "yellow"))
scatter3D(kmeans_result_2$a, kmeans_result_2$b, kmeans_result_3$c, bty = "g", pch = 18,
          col.var = as.integer(kmeans_model_1$cluster),
          col = c("red", "green", "yellow", "blue"))
scatter3D(kmeans_result_3$a, kmeans_result_2$b, kmeans_result_3$c, bty = "g", pch = 18,
          col.var = as.integer(kmeans_model_1$cluster),
          col = c("red", "green", "yellow","blue", "purple"))
scatter3D(kmeans_result_4$a, kmeans_result_2$b, kmeans_result_3$c, bty = "g", pch = 18,
          col.var = as.integer(kmeans_model_1$cluster),
          col = c("red", "green", "yellow","blue", "purple", "black"))
```